**5. (20 points) Mutual Exclusion and Synchronization**

Consider the new project revealed by Amzon.com that delivers packages using drones. Lets see if we can help amazon by organizing their fleet of drones. The drones are powered up and ready to go. Each drone has to **wait** to get (i) the **address** and (ii) the **package**. Next they all fly out of a **single drone port** (**droneOut**) out of the amazon warehouse. Thus the drone port is can handle only one drone at any time. Once the package is delivered drones come back home through another port (**droneIn**) that can also handle/allow only one drone at a time. Assume amazon drones are powered by XINU. Use appropriate **xinu semaphores** to bring order to the amazon drone zone. Clearly show the creation, initialization, usage of the semaphores, in a pseudo code for the operation of the drones.

(Hint: you need 4 semaphores some for synchronization and some for mutual exclusion).

**3. (20 points) Signal and handlers**

Consider an ECU (embedded control unit) with the following data structures and tasks. Write the C code or pseudo code for the system executing on this ECU:

1. It has a queue of “base tasks” that are aperiodic; define this as BQ in your pseudo code
2. It has a “cyclic executive schedule” of period tasks A, B and C; define this as PQ in your pseudo code
3. It has a scheduler task that gets interrupted by a **signal number 56** periodically (**every 5ms) by the system timer.**
4. The scheduler invokes a **signal handler** every time **signal 56** occurs
5. The signal handler for **signal 56** schedules the next task from the period task queue (PQ). If there is no periodic task then it invokes a base task from the base task queue (BQ).
6. After the initiating either task from PQ or BQ, the **signal 56** is armed (set up) again for receiving the signal.

Write the pseudo code for the scheduler including the signal handler, signal setup (arming the signal), timer and the related data structures. You may use dummy functions wherever needed.

**6. (20 points) Concurrent tasks and communication using pipes**

a) Draw the tasks and the pipes between them as described by the following code segment.

b) Draw the file/pipe descriptor table for the parent and the child with the relevant points indicated.

c) What is the output?

Also show the output, pipe structures and table structure at the point indicated: #1 ….. #4

int main() {

pid\_t newpid;

int fd1[2], fd2[2];

char m1[] = "Final Exam 2013\n";

char m2[] = "Today is 10th December\n";

char rbuf1[256];

char rbuf2[256];

int cn1, cn2;

if ((pipe(fd1)== -1)) printf(" error \n");

printf("fd1 %d %d \n", fd1[0], fd1[1]);

dup(fd1[0]);

dup(fd1[1]); //#1 show the open file descriptor table here

write(4, m1, sizeof(m1));

cn1 = read(5, rbuf1, 256);

write(1, rbuf1, cn1); //#2 show the structure of pipe fd1 with the respect the current process

if ((pipe(fd2)== -1)) printf(" error \n");

printf("fd1 %d %d \n", fd2[0], fd2[1]);

if ((newpid = fork()) == -1) { printf("error \n"); return 0;}

if (newpid >0 )

{ //parent

write(4, m2, sizeof(m2)); //#3 show the open file descriptor table here

}

else { //child

cn2 = read(3, rbuf2, 256);

write(1, rbuf2,cn2);

int fd3[2];

if ((pipe(fd3)== -1)) printf(" error \n");

printf("fd1 %d %d \n", fd3[0], fd3[1]); //#4 Show the open file descriptor table for parent + child

// other code

}

return 0; }}

1. **(Process management syscalls: 25 points)**
2. // filename: forkExample2.c
3. main()
4. {
5. int val1;
6. int val2;
7. val1 = fork(); /\* create new process\*/
8. if (val1 == 0)
9. {
10. execl("/bin/ls", "ls", (char \*)0);
11. printf("Message 3: \n");
12. }
13. else if (val1 > 0)
14. {
15. val2 = wait((int \*)0);
16. printf("Message 2: %d %d \n", getpid(), val2);
18. }
19. else
20. printf("Message 1: %d \n", val1);
21. return 0;
22. }
23. What is the output of the program when the program forkExample2.c is compiled and executed successfully? Assume it compiles correctly and it is the only the files associated with this program is in the directory. PIDs of parent and child are: 123, 126 respectively. We compile using

cc –o fork2 forkExample2.c

1. Under what condition is Message 3 printed out? Message 2? Message 1?

**6.** (20 points) **Finite state machine: Table driven coding**

Craps is a popular game of dice. The rules are as follows:

Phase 1:

1. You roll two dice with each die having number from 1-6
2. If the sum is 7 or 11 in the first throw you win, exit
3. If the sum is 2, 3 or 12 in the first throw you lose, exit
4. Anything else save the sum as “points” go to Phase 2

Phase 2:

1. Roll the dice
2. If you rolled a 7 you lost, exit
3. If you rolled for a sum equal to “points” of phase 1 the you win, exit
4. If you rolled for a sum not equal to points then repeat Phase 2.
5. (10 points) Draw the finite state machine (state diagram: NOT FLOW CHART) to represent the craps game.
6. (10 points) Derive a state table based on the state diagram in (a).

4. **(20 points) WRT54GL, Embedded XINU, Projects**

a) What is a cross-compiler? How is it used for working with embedded XINU environment?

b) What does function call **create** accomplish in the following command?

**ready(create((void \*)shell, INITSTK, INITPRIO, "SHELL0", 1, CONSOLE),RESCHED\_NO);**

**c)** what does function call **ready** accomplish using the following command?

ready(create((void \*)shell, INITSTK, INITPRIO, "SHELL0", 1, CONSOLE),RESCHED\_YES);

**d)** What does qualifier “volatile” for a variable indicate? In other words when will you use this modifier? Where was it used in Xinu?

e) Explain the important characteristic of (i) automatic (ii) dynamic and (iii) static variables?

1. Memory management in RTEM systems

Explain 5 common pitfalls when allocating and using memory in RTEM systems. Use code examples.

1. Use case analysis of your